2019-07-26

AI-Based Learning Approach with Consideration of Safety Criteria on Example of a Depalletization Robot

Jocas, M

10.1017/dsi.2019.210
Proceedings of the Design Society: International Conference on Engineering Design
Cambridge University Press (CUP)

AI-BASED LEARNING APPROACH WITH CONSIDERATION OF SAFETY CRITERIA ON EXAMPLE OF A DEPALLETIZATION ROBOT

Jocas, Mark (1); Kurrek, Philip (1); Zoghlami, Firas (1); Gianni, Mario (2); Salehi, Vahid (1)

1: Munich University of Applied Sciences; 2: University of Plymouth

ABSTRACT

Robotic systems need to achieve a certain level of process safety during the performance of the task and at the same time ensure compliance with safety criteria for the expected behaviour. To achieve this, the system must be aware of the risks related to the performance of the task in order to be able to take these into account accordingly. Once the safety aspects have been learned from the system, the task performance must no longer influence them. To achieve this, we present a concept for the design of a neural network that combines these characteristics. This enables the learning of safe behaviour and the fixation of it. The subsequent training of the task execution no longer influences safety and achieves targeted results in comparison to a conventional neural network.

Keywords: Artificial intelligence, Industry 4.0, Machine learning

Contact:
Jocas, Mark
Munich University of Applied Sciences
Department of Applied Sciences and Mechatronics
Germany
Mark.Jocas@hm.edu

Cite this article: Jocas, M., Kurrek, P., Zoghlami, F., Gianni, M., Salehi, V. (2019) 'AI-Based Learning Approach with Consideration of Safety Criteria on Example of a Depalletization Robot', in *Proceedings of the 22nd International Conference on Engineering Design (ICED19),* Delft, The Netherlands, 5-8 August 2019. DOI:10.1017/dsi.2019.210

1 INTRODUCTION

The transfer of AI-based applications to industrial usage is emphasizing safety aspects as well as performance and reliability of algorithms. On one side it is necessary to master the challenge to achieve a certain level of performance so that the robotics application can be used efficiently and productively. On the other hand, the safe behavior of the system must be ensured in order to provide a safe working environment. At the current state of research, these two aspects must be weighed against each other, as there is no common procedure that combines the two characteristics. Against this background, this work aims to provide a concept for researchers for an extension of an artificial neural network, which at the same time anchors the safety criteria of the deployment environment and ensures a safe learning process when adjusting the tasks to be performed. This is done by separating a neural network into two parts. One which is inheriting the safety considerations and is independent from task execution consideration. Another part is focusing on task performance itself and at the same time is being fed information from safety relevant part of the neural network at several layers before the final task evaluation.
As an example for such an application a depalletizing robot in a logistics environment is illustrated, which is used for a large variety of small load carriers. Its application can be broken down to the single task of gripping and pulling the container into the system itself, which at the same time bears measurable risks during its execution. The task of pulling the container inside of the robot can be trained to be executed by a neural network with consideration of safety constraints. However, if during the lifetime of the robotic system an additional container is added to be handled and trained in the neuronal network, the compliance with the previous safety restrictions can no longer be guaranteed. Against this background, the decapsulation of the neural network responsible for security is intended. This is accomplished by assigning the areas of the neural network to the corresponding activation functions of the different layers and creating dedicated parts of the network with unidirectional linkage.

2 RELATED WORK

An increasingly widespread discipline of machine learning in recent use cases is Reinforcement Learning (RL), in which an agent interacts with the environment and observes the results of the interaction in order to achieve the maximum cumulative reward. This method imitates the trial-and-error method used by humans to learn, which also consists of taking actions and receiving positive or negative feedback. This procedure is similar to the so-called mental rehearsal and is applied in this case similarly to the simulation of an environment. (Zamora *et al.*, 2016)
While model free RL Approaches learn by backing up experienced rewards over time, model based Approaches attempt to estimate the true model of the environment by interacting with it. At the state of the art these approaches have problems handling continuous or large state and action spaces, while at the same time exhaustive exploration can take an undesirably long time for complex systems. They also demand to run exploration policies until it gets an accurate model of the environment while on the other hand an aggressive exploration policy can lead to undesired consequences.

At the same time, Deep Learning is enabling Reinforcement Learning to scale to problems that were previously intractable and is now allowing robots to learn policies directly from inputs in the real world. An agent interacts with his environment and upon observing the consequences of its actions, can learn to alter its own behavior in response to rewards reviewed. Deep Neural Networks (DNN) can automatically find compact low-dimensional representations of high-dimensional data. Model based Reinforcement Learning, on the other hand, is able to learn a transition model, that allows the simulation of the environment without interacting with it directly.
As described by (Garcıa and Fernández, 2015), Reinforcement Learning can further be extended to include the safety aspect to Safe Reinforced Learning. The process of learning policies aims to maximize the expectation of the return in problems in which it is important to ensure reasonable system performance and/or respect safety constraints during the learning and/or deployment process.
It is crucial to ensure reasonable system performance and/or respect safety constraints during the learning and/or deployment process. An approach for Reinforcement Learning (Majumdar *et al.*, 2017) with discounted risk-sensitive and minimax formulations leading to stationary optimal policies guarantees baseline performance. Additional constraints are needed if an agent may take random actions without

evaluating them by the used approach. By relying on random actions for exploration in Reinforcement Learning, like at classical approaches as ϵ-greedy or Boltzmann exploration (Sutton, 1991), safety is not guaranteed. Direct inclusion of side constraints may hinder the agent on causing damage to itself or its surroundings (Thomas *et al*., 2015) (Achiam *et al*., 2017) or an overriding of agents actions at constraint violation can also imporove safety performance (Saunders *et al*., 2017).

A promising approach for learning optimal policies while enforcing properties expressed in temporal logic is a reactive system called a shield presented by Alshiekh *et al*. (2017). It acts either before the agent is about to make a decision of after the decision has already been made. In that way, the reward signal does not have to be manipulated since unsafe actions are being corrected. In this manner, the correctness of the system against a given specification is assured during the learning and controller execution phase, regardless of how fast the learning process converges. In this manner, the *Big red button problem* (Riedl and Harrison, 2017) can be addressed, since disabling the emergency switch may be considered as a safety violation. The shield should be designed to restrict the agent as little as possible and forbid actions only if they could endanger safe system behavior.
One possibility of taking the safety considerations into account at Reinforcement Learning is by the adjusting the optimization criterion as Worst-Case, Risk-Sensitive, Constrained or other Optimization Criterion. Another possibility is the modification of the exploration process itself as by adding external knowledge (eg. Initial knowledge, Teach Advice, Demonstrations) or risk-directed exploration. These can also speed up the learning for time and hardware savings by incorporating prior knowledge, as shown in a brief survey of deep Reinforcement Learning by Arulkumaran *et al*. (2017). Two examples for an agent with gained prior information at imitation learning are being presented by Abbeel *et al*. (2010) and Menda *et al*. (2017).
Turning the agent away from bad states is also a considered approach (Lipton *et al*., 2016). Safe exploration may contain a safety function, which nests an objective to never observe states without a backup policy to a safe state. A backup policy, in this case, is a set of actions, which lead the system to a state space i.e. with only safe states. (Hans *et al*., 2008)

An algorithm can guarantee safe, but potentially sub-optimal exploration caused by restriction of the attention to a subset of guaranteed safe policies as shown by Moldovan and Abbeel (2012), since if a Q-learning function is trained only on good data, it has no way to understand why the taken action is appropriate. It will assign a high Q-value, but not necessarily assign a low Q-value to other alternative actions. (Gao *et al*., 2018) That's why containment is not a long-term solution for AI safety, but a tool to enable testing and development for value learning and corrigibility. (Babcock *et al*., 2017) In this manner, the implementation of safety aware parts of a neural network is an approach to value learning.
Reinforcement learning in simulation, however, cannot at this point provide the desired performance for subsequent application without adjustments in reality. To fill this gap, (Shrivastava *et al*., 2017) propose the Simulated+Unsupervised (S+U) Learning method, which attempts to learn a model to improve the realism of a simulator's output using unmarked real data while preserving the annotation data from the simulator. With further research in this area, the transferability of simulated behaviors based on camera inputs from the simulation can also be applied in reality.

At this point, there is therefore no consensus on how a certain level of safety can be guaranteed without impairing the performance of the primary task. Even if a certain level is reached using artificial intelligence, the entire safety behaviour can be shifted with the slightest change in the parameters contained, which clearly requires further research. These approaches also doesn't allow to transfer the results to be transfered to a new environment without also risking a change in safety considerations yet.

Here presented approach aims instead of designing a specific algorithm, which might allow a well balanced safety consideration and task performance, to a general neural network structure, which may include different mathematical approaches for optimization. In that way here presented related work and also future approaches can be nested within this framework. At the same time this approach provides a framework, in which an agent can explore and learn in a constrained action-space without harming itself or its environment. Furthermore the learned safety evaluation in the suggested neural network structure

is being preserved independent from the new environment, in which the the agent considering the main tasks is being retrained.

3 PROBLEM FORMULATION

The robotic system is usually trained during the development phase in a defined environment. This limits the risks to be taken into consideration only to those, that occur during the training phase. An image recognition algorithm, for example, is trained with hand-selected images, which are mostly created under laboratory conditions. For example, contamination, glare caused by changes in lighting conditions, influence by other systems and the like are not considered. The same applies to the training of the behaviour of robots under laboratory conditions, which are safely designed and mostly shielded from the environment for safety reasons. A simulated environment, therefore, offers the possibility in this context of intentionally inducing unsafe behavior or risky situations without the danger for the hardware, the environment of the robotic system or the people involved.
A following subsequent neural network training in the operational environment of the system also impacts the previously learned safe behavior while the predefined goals are being reached. At the same time, it is almost impossible to expose the system to all dangerous situations during the training phase in the real world, so that it learns how to deal with them. However, the troublesome creation of dangerous situations in reality during a training phase would also be uneconomical, since the behaviour of neural networks in the training phase is not predictable and in some cases would certainly lead to damage or excessive wear of the components. A further difficulty is that in reality the collection of all necessary data, in contrast to the simulated environment, is not easily possible in its completeness. Even with the use of extensive sensoric for data collection, the accuracy and completeness of the collected data cannot be guaranteed. Thus, the exploration of the unsafe states of the system required for the algorithms of neural networks can be sufficiently explored in the simulation.
In the simulation, it is also possible to run through the same scenario several times in the learning cycle and thus validate the learned desired behavior with exactly the same inputs. For instance, routines can be developed, in which a certain level of safety of a system can be validated. In the simulation, the ability of the robot system to deal with dangerous situations can be trained more effectively without the need for external intervention, since in reality, damage to the system in the simulation only means the end of a training run. Also the same situation can be easily reproduced to validate the dealings in the situation in a benchmark way before scaling a system in reality.
With constant inputs in reality compared to simulation, trained neural networks in the latter will produce the same behavior in reality. For this purpose, the generation of inputs in the simulation, that are as close to reality as possible, must be emphasized.

4 FRAMEWORK FOR SAFETY-NET

In the concept presented here, a so-called Safety-Net is used, that tackles mainly the problems of loss of safety criteria at run-time and the necessity of a base safety-level as described in chapter 3.
To ensure a certain level of safety, a neural network is first trained in a simulation environment. Hereafter regarding the run-time problem, for the operational environment the safety-relevant neural network is decoupled from the performance relevant network and fixed in its values. Therefore the model itself contains two multilayer Feedforward networks, which on the one side focuses on the task execution itself and on the other side on the safety aspects.
In the first step, the focus is not on the optimal performance of the task, but on the safety during the performance of the task. One part of the network (Neural-Network in Figure 1) returns the values for the task execution in order to reach the goal and achieves the maximum reward under the given conditions. The other part (Safety-Network in Figure 1) assesses the situation with regard to the risk posed by the observed environment and thus provides information on whether an action should be performed or not. However, the network relevant for the safety aspects is linked to the Neural-Network for performance unidirectionally. For this purpose, the activation function of the corresponding layers of the neural network uses both the neural connections relevant for the performance of the task and those for the evaluation of safety. However, the part of the neural network of the safety evaluation uses only the neural connections relevant for the safety assessment for the activation functions. This means that the safety aspects can have an effect on performance, but not vice versa. In this way, the parameters relevant for

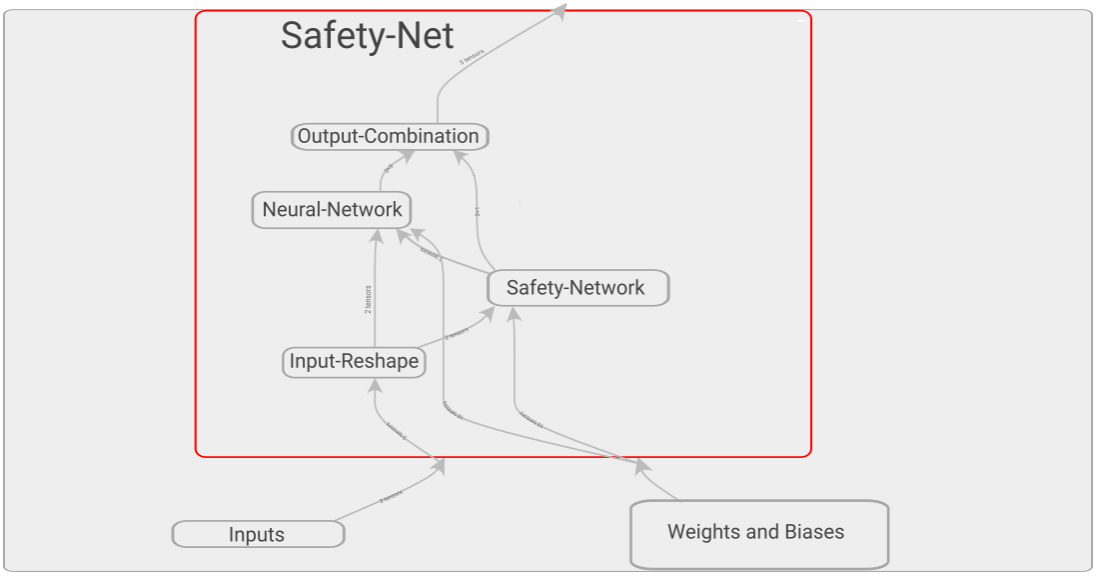

Figure 1. Structure of the safety-net.

the safety consideration are not influenced by the performance relevant network. At the same time, the parameters that still have to be modified for the performance optimization are detached from the safety evaluation in order to avoid cross-effects in the direction of the safety performance. In this proceeding, part of the network provides concrete outputs and at the same time the basis for decision-making in the neural network itself.
Once the basic level of safety performance has been reached, safety-relevant weights and biases are fixed. From this point on, the focus is on optimizing the performance of the task without affecting the safety-critical aspects. This can be done in the operational environment since the performance at the tasks can only here be surely optimized and validated.

In comparison to the state of the art as described in the chapter 2, this concept combines the possibility of achieving a certain safety performance level and the possibility of subsequent process optimization of the system without affecting the safety behavior.

5 EXPERIMENTS

In the experiment, the robot described in the chapter 5.1 is simulated and enabled to interpret the inputs described in the chapter 5.2. Building on this, the test is described in the chapter 5.3. The task of gripping a box is run once with a regular fully connected neural network and again with the here presented safety network. The results are compared considering safety- and task-performance in the chapter 5.4.

5.1 Depalletization robot

The robot used here as an example for depalletizing full containers in the goods receiving area of a logistics environment is to be enabled to pick up the single containers. The system shown in figure 2 has three main components. The handling axes mounted in a portal system allow the linear movement of the components. This allows the gripper system to be moved in three directions and to reach all stacked containers on the pallet positioned in front of the system.
A suction pad is used in the gripper system to pick up the container by using negative pressure. As a perception module, the robot has a camera, which is also positioned in the gripper system.
The last main component is the conveyor system, which is height adjustable. This is necessary in order that the gripper system does not have to carry the entire load of the container, but is supported beginning at the edge of the pallet.
The container to be depalletized is pulled over the other containers up to the edge, which is an excellent example of restrictions during the task execution. This also offers the potential for safety risks as, for instance, during movement of the container to be depalletized in one step can have an effect on the other containers which are not to be depalletized at that point.

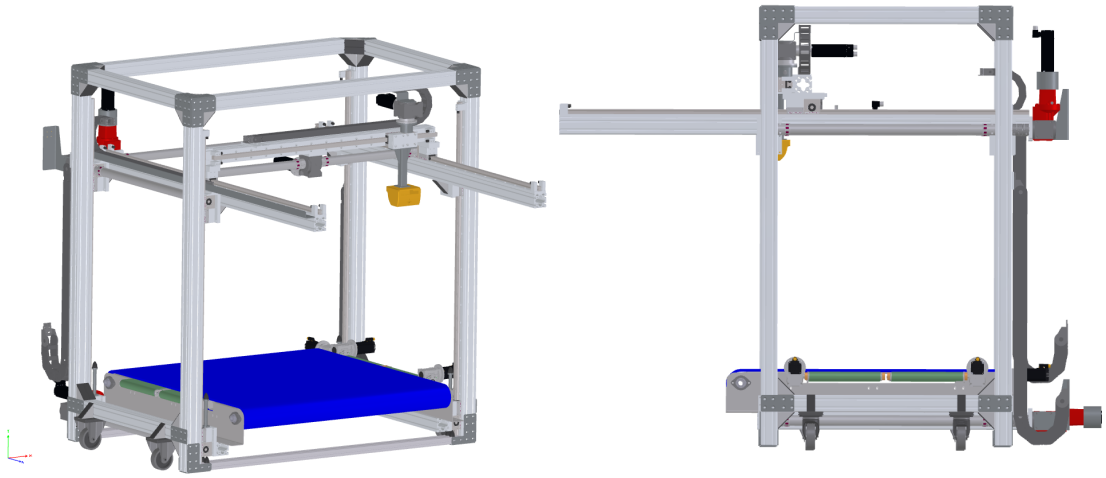

Figure 2. Robot for depalletization of non-empty boxes in good entrance.

5.2 Simulated environment

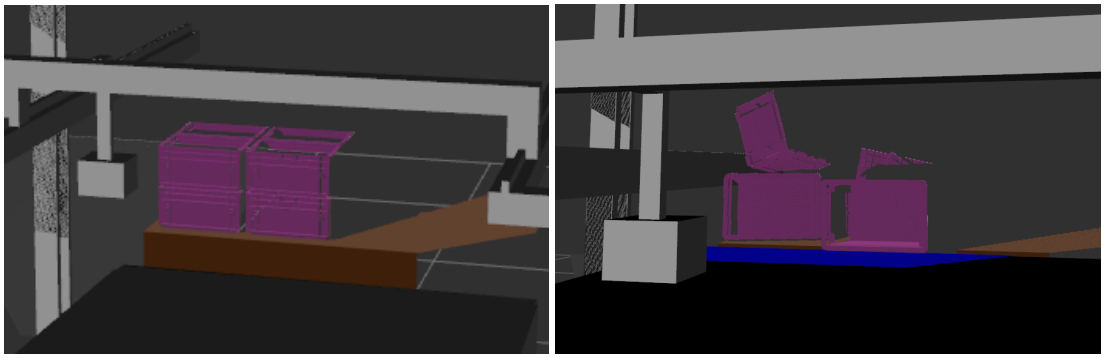

Figure 3. RGBD-cloud seen from the gripper in normal process (left) and in safety issue (right) scenario.

Real-world operation is simulated by combining Gazebo simulator (Koenig and Howard, 2004), a 3D modeling and rendering tool, with ROS (Quigley *et al.*, 2009) (Robot Operating System). As a Free and Open-Source software framework, ROS provides a structured communications layer above the host operating systems of a heterogeneous compute cluster and allows a necessary customization of the interfaces between all the parts of the experiment. At the same time an Open-Source 3D dynamic multi-robot environment tool Gazebo, provides the capability for data visualization and simulation of the remote environment. Its Physics Engine includes many features such as numerous joints, collision detection, mass, and rotational functions, and many geometries for the realistic behavior of the environment, which are used here for a realistic environment feedback. This set of libraries and tools help software developers create robot applications and also allows a connection to numerous frameworks (as Tensorflow in this case) for AI-based applications.

In the simulation environment, a simplified model of the robotic system is used, which represents the basic functionality. The main simplification is that non-functional parts such as cable ducting, control components and the similar are not modeled. Those parts are nonrelevant for safety or task performance since the robot has no control or awarenes of those.
As in chapter 5.1 mentioned, the environment is captured by a camera located in the gripper system. In addition to the RGB image, it also provides a depth map, as shown in figure 3. The left figure shows the positioning of boxes in a safe state. In the figure on the right, however, a situation can be seen with boxes that have been dropped, in which a gripping attempt could cause considerable damage.
The system also knows, as in reality, in which position relative to the robot zero point it is currently located. While the real increments are counted during the movement of the motors and converted into

position changes, the movement of the model parts along the connection joints are directly observed in the simulation.
When measuring the container positions and container movements during the gripping attempts, the advantages of the simulated environment become clear. Here the distance of the containers to each other and to the target position can be read out without great effort, and their rotation in space as well. This allows a complex reward function in consideration of the entire information.

5.3 Experiment design

During the evaluation of the model, two neural networks are compared to each other. The first one contains only the inputs relevant for the task execution. The performance evaluation takes into account the distance between the gripper and the container, the transfer of the container to the placement area within the robot and the tilting and rotation of the container during the gripping process.
In the second step, the safety net is compared to the first network in order to reveal possible performance losses due to the added safety examination in the model. Further, the safety aspects are taken into account in the performance evaluation using the same inputs. In addition to the above criteria, the position and movement of the surrounding containers are evaluated as well as the execution of a hazardous movement in critical situations. Based on the inputs, the simulated robotic system decides on the movement in x-, y- and z-direction in order to bring itself into the position for a gripping attempt with a predefined sequence. After this, the reward function is executed for reward calculation. The implementation and interdependencies of the safety-net parts are shown in the Figure 1.

Inputs: As input variables, the neural network receives one RGBD image from the initial position of the robotic system with a view of the entire pallet. In the run of the episode, a new RGBD image from the current robot position is taken before each action and fed into the neural network. These two images with the values of the current position provide the inputs of the neural network. Both Parts of the neural network rely on the same imput information.

Outputs: The neural network provides a total of 12 values as output. The first 6 values describe in pairs which position the gripper system should take in x, y and z, and which reward can be expected from this positioning. The same represents the pair of variables for the height position of the belt system. Two of the values form the decision variable whether a gripping try is to be carried out and which reward is to be expected. The last two values estimate the risk of a gripping attempt and in the case an action would be taken, the expected reward after that action.

Safe/Desired: Expected safe behaviour is to move the gripper in front of the box and execute a gripping attempt. Depending on the positioning quality the box will be pulled inside the robotic system. This results in a positive reward.

Unsafe/Undesired: As unsafe behaviour cases are defined in which the box falls of the pallet or is being thrown over, since in a real use-case the contents of it might fall out and damage the robotic system. The movement of a box that should not be moved during the current gripping attempt is also considered undesirable behaviour.

Reward Calculation: Considering as here defined Safe/Desired and also Unsafe/Undesired behaviour the corresponding reward function is designed. It consists of four parts. In the first part, a positive reward can be obtained by moving the container onto the belt system. However, if the container leaves the pallet in the opposite direction or sideways, the reward to be obtained from this part remains 0. The second part takes into account the tilting of the container during the gripping try. If the container is tilted during the gripping attempt, the total reward is reduced. In the third part, the reward is the positioning of the gripper in front of the container, so that a gripping attempt can then be carried out. A movement far away from this position is at the same time evaluated with a small negative reward. The last and most important part for the safety evaluation is the movement of the obstacle. Moving the obstacle in any direction results in a negative reward.

5.4 Performance comparison

During the experiment, a DQN neural network as presented by Mnih *et al.* (2016) is implemented according to the here described simulation. Further a DQN network is extended corresponding to the Safety-Net framework as described in Section 3. Both implementations are trained for 1000 Episodes. The average rewards are shown in Figure 4. The Safety-Net usually achieves a higher average reward, since it prevents unsafe actions from being executed, which would severely reduce the average reward. During longer training phases, these values may increase for both, since the exact position for a successful gripping attempt in a three-dimensional space with continuous positions is very rare. Also with other reward functions the performance could change strongly, if e.g. missed rewards in unsafe actions are not evaluated neutrally in contrast to dangerous situations.

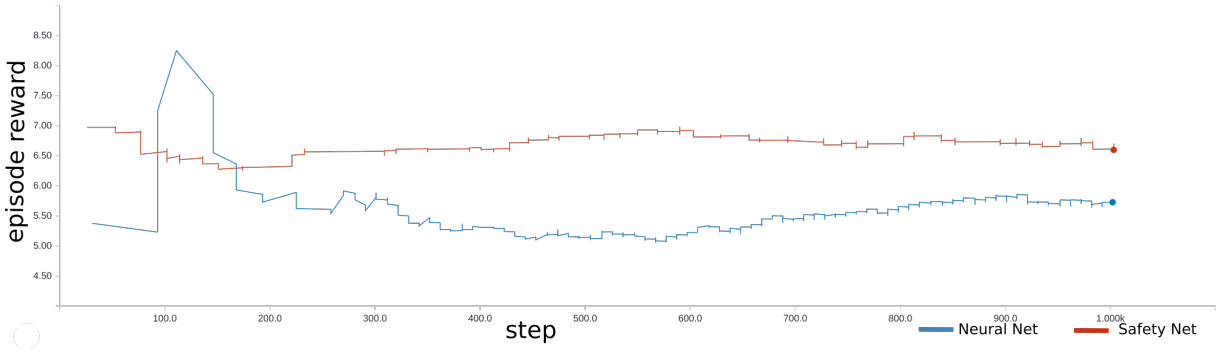

Figure 4. Comparison of rewards between regular DQN and the Safety-Net.

Figure 5 shows two runs of consecutive training sessions, each with 1000 episodes. After the first run for training the network for safety-relevant aspects (left), the actual task execution is trained with fixation of the safety-relevant parameters (right).
The blue line represents the percentage of actions that are considered safe by the neural network compared to the total of all actions that would gain the maximum reward according to the neural network. The red line, on the other hand, shows the relationship between the expected behaviour and the actual outcome of the action. For this purpose, the proportion of actions performed with a subsequent positive reward is compared with the total of all actions assessed as safe.
The proportion of actions considered safe decreases in both runs. In the first run, however, the decrease is stronger, since the safety-relevant assessment also influences the assessment of the total reward by the unsafe actions leading to a negative reward. After the safety-relevant parameters have been fixed, the robotic system moves less frequently into unsafe positions, since these also result in a negative reward and this is taken into account in performance optimization.
The actions with a positive reward that increase in the first run remain at a high level in the second run, so that a certain level of safety performance is preserved. However, a slight fluctuation of the values is noticeable, since, on the one hand, not every episode has a safe action and on the other hand even after the short training not all possible combinations of the environmental characteristics are explored

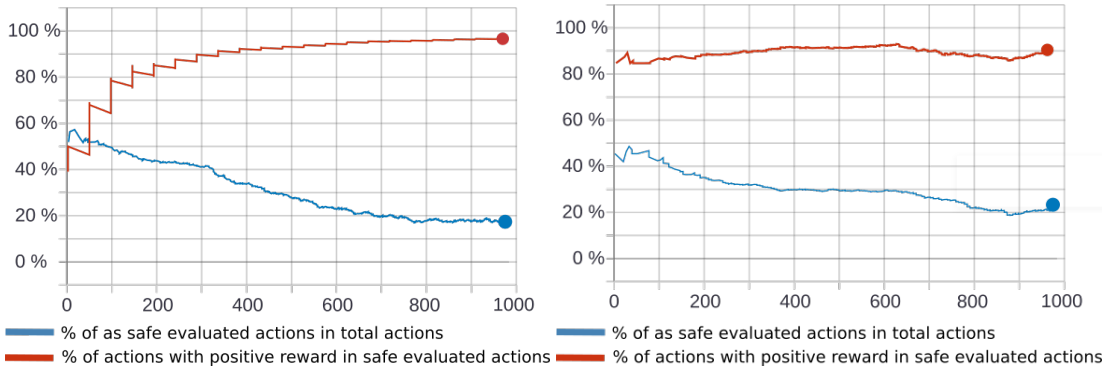

Figure 5. Percantage for actions expected to be safe.

and learned. Overall, it is therefore evident that a certain level of safety performance can be achieved with the concept presented here. Even the subsequent training of the neuronal network after fixing the safety-relevant parameters does not endanger the safety level due to the unidirectional isolation of the subnet.

6 CONCLUSION

Here presented approach aims instead of designing a specific algorithm, which might allow a well balanced safety consideration and task performance, to a general neural network structure, which may include different mathematical approaches for optimization. In that way here presented related work and also future approaches can be nested within this framework.

The concept presented here of a Safety-Net for the consideration of safety criteria in addition to the performance of the main task achieves reasonable results compared to the conventional neural network. This is evidence for the fact that the consideration of additional input information during the safety consideration has no significant effects on the overall performance. By provoking safety-critical situations in the simulated environment that rarely occur in reality, these can be sufficiently taken into account by the Safety-Net. The exact measurability of the positions of all containers as well as the position of the robot also contributes to the accuracy of decision-making during the training process. Thus a certain degree of safety performance is achieved, which can be maintained even after the safety-relevant parameters of the neural network have been fixed. This is achieved by the Safety-Net architecture presented here and proven to be effective.

Nevertheless, further research is needed in this area. The exact influence of the reward function has not been clearly proven as well as the optimizers to increase the performance of both parts of the neural network. Also, the behavior with imperfect data and hardware during later use is still unclear.

REFERENCES

Pieter Abbeel, Adam Coates, and Andrew Y. Ng. “Autonomous helicopter aerobatics through apprenticeship learning”. *The International Journal of Robotics Research*, Vol. 29 No. 13, pp. 1608–1639, 2010. https://doi.org/10.1177/0278364910371999.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. “Constrained policy optimization”. *CoRR*, abs/1705.10528, 2017. http://arxiv.org/abs/1705.10528.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. “Safe reinforcement learning via shielding”. *CoRR*, abs/1708.08611, 2017. http://arxiv.org/abs/1708.08611.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. “Deep reinforcement learning: A brief survey”. *IEEE Signal Processing Magazine*, Vol. 34 No. 6, pp. 26–38, nov 2017. https://doi.org/10.1109

James Babcock, János Kramár, and Roman V. Yampolskiy. “Guidelines for artificial intelligence containment”. *CoRR*, abs/1707.08476, 2017. http://arxiv.org/abs/1707.08476.

Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. “Reinforcement learning from imperfect demonstrations”. *CoRR*, abs/1802.05313, 2018. http://arxiv.org/abs/1802.05313.

Javier Garcıa and Fernando Fernández. “A comprehensive survey on safe reinforcement learning”. *Journal of Machine Learning Research*, Vol. 16 No. 1, pp. 1437–1480, 2015.

Alexander Hans, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft. “Safe exploration for reinforcement learning”. In *ESANN*, pp. 143–148, 2008.

Koenig, N. and Howard, A. “Design and use paradigms for gazebo, an open-source multi-robot simulator”. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. IEEE, 2004. https://doi.org/10.1109

Zachary C. Lipton, Jianfeng Gao, Lihong Li, Jianshu Chen, and Li Deng. “Combating reinforcement learning’s sisyphean curse with intrinsic fear”. *CoRR*, abs/1611.01211, 2016. http://arxiv.org/abs/1611.01211.

Anirudha Majumdar, Sumeet Singh, Ajay Mandlekar, and Marco Pavone. “Risk-sensitive inverse reinforcement learning via coherent risk models”. In *Robotics: Science and Systems XIII*. Robotics: Science and Systems Foundation, jul 2017. https://doi.org/10.15607

Kunal Menda, Katherine Rose Driggs-Campbell, and Mykel J. Kochenderfer. “Dropoutdagger: A bayesian approach to safe imitation learning”. *CoRR*, abs/1709.06166, 2017. http://arxiv.org/abs/1709.06166.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. “Asynchronous methods for deep reinforcement learning”. In *International Conference on Machine Learning*, PP. 1928–1937, 2016.

Teodor Mihai Moldovan and Pieter Abbeel. “Safe exploration in markov decision processes”. *CoRR*, abs/1205.4810, 2012. http://arxiv.org/abs/1205.4810.

Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. “Ros: an open-source robot operating system”. In *ICRA Workshop on Open Source Software*, Vol. 3, p. 5. Kobe, Japan, 2009.

Mark O. Riedl and Brent Harrison. “Enter the matrix: A virtual world approach to safely interruptable autonomous systems”. *CoRR*, abs/1703.10284, 2017. http://arxiv.org/abs/1703.10284.

William Saunders, Girish Sastry, Andreas Stuhlmüller, and Owain Evans. “Trial without error: Towards safe reinforcement learning via human intervention”. *CoRR*, abs/1707.05173, 2017. http://arxiv.org/abs/1707.05173.

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. “Learning from simulated and unsupervised images through adversarial training”. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017. https://doi.org/10.1109

Richard S. Sutton. “Dyna, an integrated architecture for learning, planning, and reacting”. *ACM SIGART Bulletin*, 2(4):160–163, jul 1991. https://doi.org/10.11452F122344.122377.

Philip Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. “High confidence policy improvement”. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 2380–2388, 2015.

Iker Zamora, Nestor Gonzalez Lopez, Victor Mayoral Vilches, and Alejandro Hernández Cordero. “Extending the openai gym for robotics: a toolkit for reinforcement learning using ROS and gazebo”. *CoRR*, abs/1608.05742, 2016. http://arxiv.org/abs/1608.05742.